US012399638B2

(12) **United States Patent**
Gorobets et al.

(10) **Patent No.: US 12,399,638 B2**
(45) **Date of Patent: Aug. 26, 2025**

(54) **TLC DATA PROGRAMMING WITH HYBRID PARITY**

(71) Applicant: **Sandisk Technologies, Inc.**, Milpitas, CA (US)

(72) Inventors: **Sergey Anatolievich Gorobets**, Edinburgh (GB); **Alan D. Bennett**, Edinburgh (GB); **Liam Parker**, Edinburgh (GB); **Yuval Shohet**, Acton, MA (US); **Michelle Martin**, Westford, MA (US)

(73) Assignee: **Sandisk Technologies, Inc.**, Milpitas, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 15 days.

(21) Appl. No.: **18/515,758**

(22) Filed: **Nov. 21, 2023**

(65) **Prior Publication Data**

US 2024/0086097 A1 Mar. 14, 2024

Related U.S. Application Data

(62) Division of application No. 17/202,163, filed on Mar. 15, 2021, now Pat. No. 11,861,195.

(51) **Int. Cl.**
G06F 3/06 (2006.01)
G06F 11/10 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 3/064*** (2013.01); ***G06F 3/0626*** (2013.01); ***G06F 3/0673*** (2013.01); ***G06F 11/1004*** (2013.01)

(58) **Field of Classification Search**
None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

8,279,668 B2 10/2012 Cho et al.
8,499,215 B2 7/2013 Kong et al.
(Continued)

FOREIGN PATENT DOCUMENTS

CN 105359218 A 2/2016
CN 107403646 A 11/2017
(Continued)

OTHER PUBLICATIONS

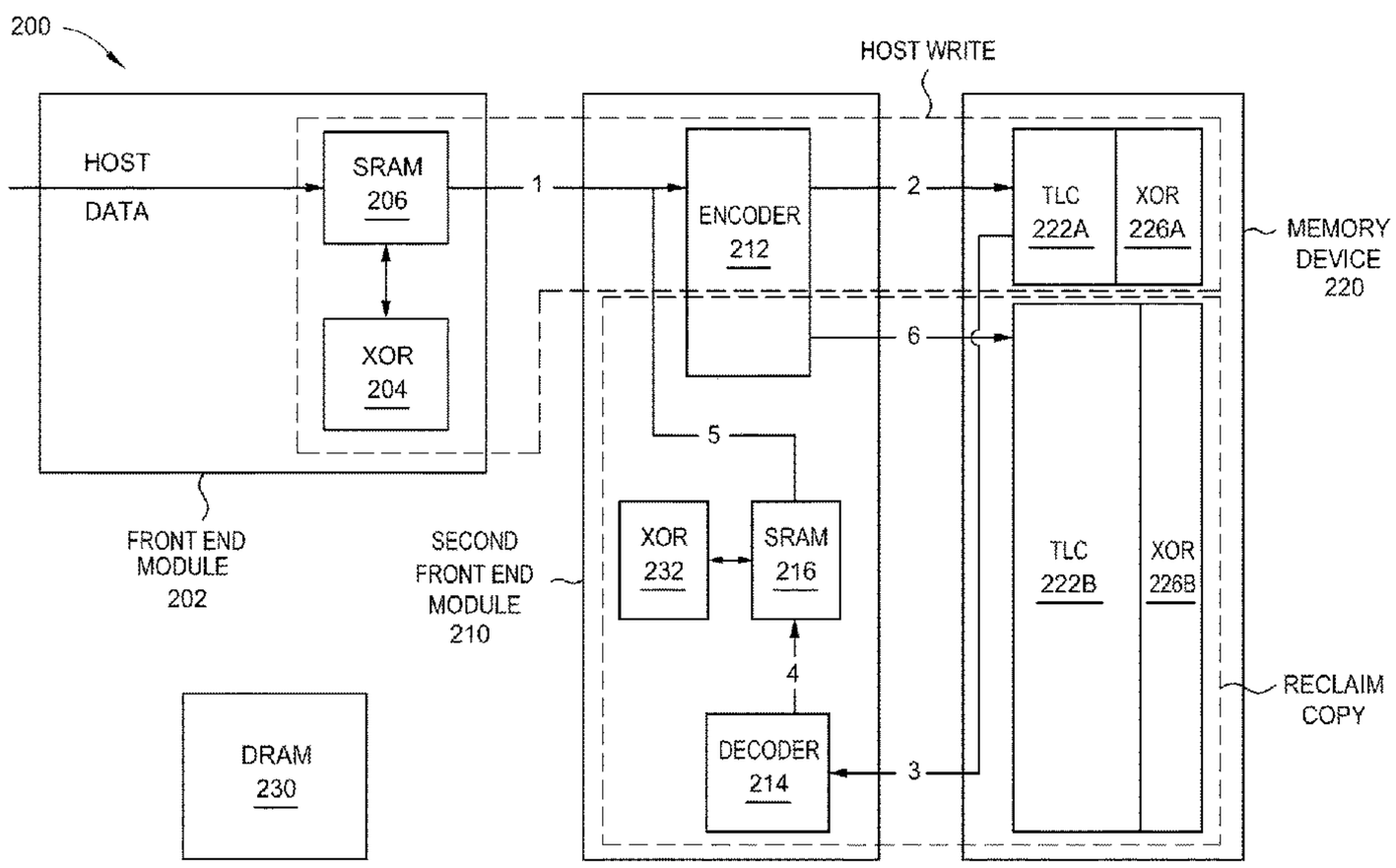
Benoit et al. "Efficient Checkpoint/Verification Patterns for Silent Error Detection," ICL Research Report, RR-1403, May 2014, 16 pages.
(Continued)

Primary Examiner — Brian R Peugh
(74) *Attorney, Agent, or Firm* — PATTERSON + SHERIDAN, LLP

(57) **ABSTRACT**

The present disclosure generally relates to improving programming to data storage devices, such as solid state drives (SSDs). A first memory device has a first XOR element and a second memory device has a second XOR element. The ratio of the first XOR element to the capacity of the first memory device is substantially smaller than the ratio of the second XOR element to the capacity of the second memory device. A read verify operation to find program failures is executed on either a wordline to wordline basis, an erase block to erase block basis, or both a wordline to wordline basis and an erase block to erase block basis. Because the program failures are found and fixed prior to programming to the second memory device, the second XOR element may be decreased substantially.

20 Claims, 12 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

8,775,901	B2	7/2014	Sharon et al.	
8,898,374	B2	11/2014	Yang	
8,902,652	B1	12/2014	Huang et al.	
9,135,155	B2	9/2015	Sharon et al.	
9,619,321	B1	4/2017	Haratsch et al.	
10,025,662	B2	7/2018	Yang et al.	
10,101,942	B1	10/2018	Parker et al.	
10,204,007	B2	2/2019	Zhang et al.	
10,339,000	B2	7/2019	Yang et al.	
10,769,014	B2	9/2020	Cariello	
10,789,126	B2	9/2020	Singidi et al.	
11,309,051	B2	4/2022	Asami et al.	
2012/0008389	A1	1/2012	Kim	
2017/0315867	A1	11/2017	Yang et al.	
2018/0025776	A1	1/2018	Achtenberg et al.	
2019/0354478	A1*	11/2019	Kashyap	G06F 11/1072
2020/0110661	A1	4/2020	Singidi et al.	
2020/0183779	A1	6/2020	Cariello	
2020/0194064	A1	6/2020	Barndt et al.	

FOREIGN PATENT DOCUMENTS

CN	109767806	A	5/2019
CN	110928722	A	3/2020
CN	111415699	A	7/2020
CN	111710358	A	9/2020
EP	2513907	A1	10/2012
WO	2011075597	A1	6/2011
WO	2017087059	A2	5/2017

OTHER PUBLICATIONS

Cai et al. "Error characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives," Proceedings of the IEEE, vol. 105, No. 9, pp. 1666-1704, Sep. 2017, doi: 10.1109/JPROC.2017.2713127, <https://arxiv.org/pdf/1706.08642.pdf>.

International Search Report and the Written Opinion for International Application No. PCT/US2020/066420 mailed Apr. 7, 2021, 11 pages.

Huilong, Y., et al., "NAND flash memory large capacity image stored invalid block management", Microelectronics and Computer, Issue 2, 2010.

Xinfeng, Y., et al., "Design of Low Error Rate Digital Telemetry System for Reentry Vehicle", Beijing Aerospace Long March Aircraft Research Institute, vol. 8, No. 7, 2020.

Zuolo, L., et al., "LDPC soft decoding with improved performance in 1X-2X MLC and TLC Nand flash-based solid state drives", IEEE, vol. 7, Issue 3, 2019.

* cited by examiner

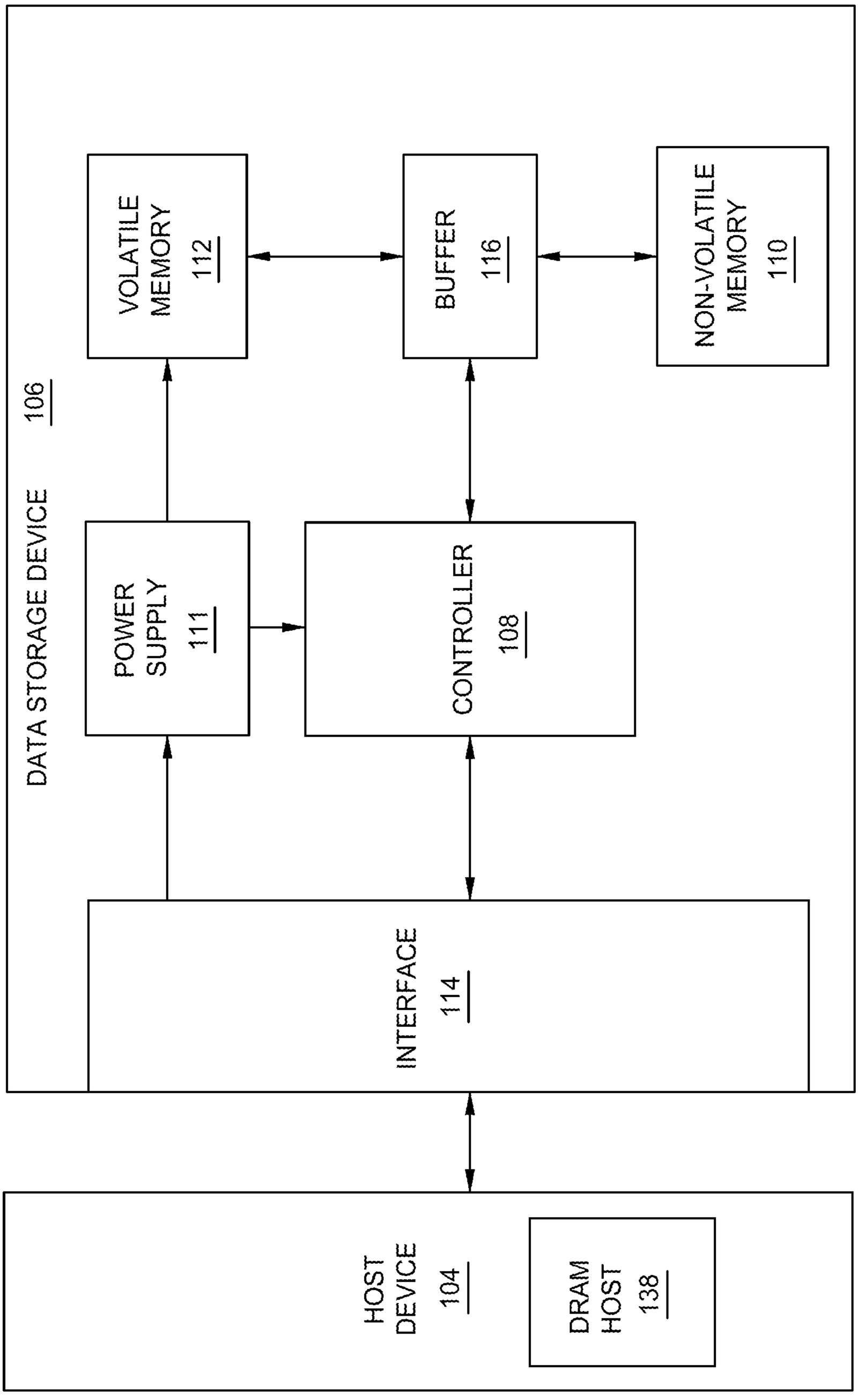

100
HOST DEVICE 104
DRAM HOST 138
DATA STORAGE DEVICE 106
INTERFACE 114
POWER SUPPLY 111
CONTROLLER 108
VOLATILE MEMORY 112
BUFFER 116
NON-VOLATILE MEMORY 110

FIG. 1

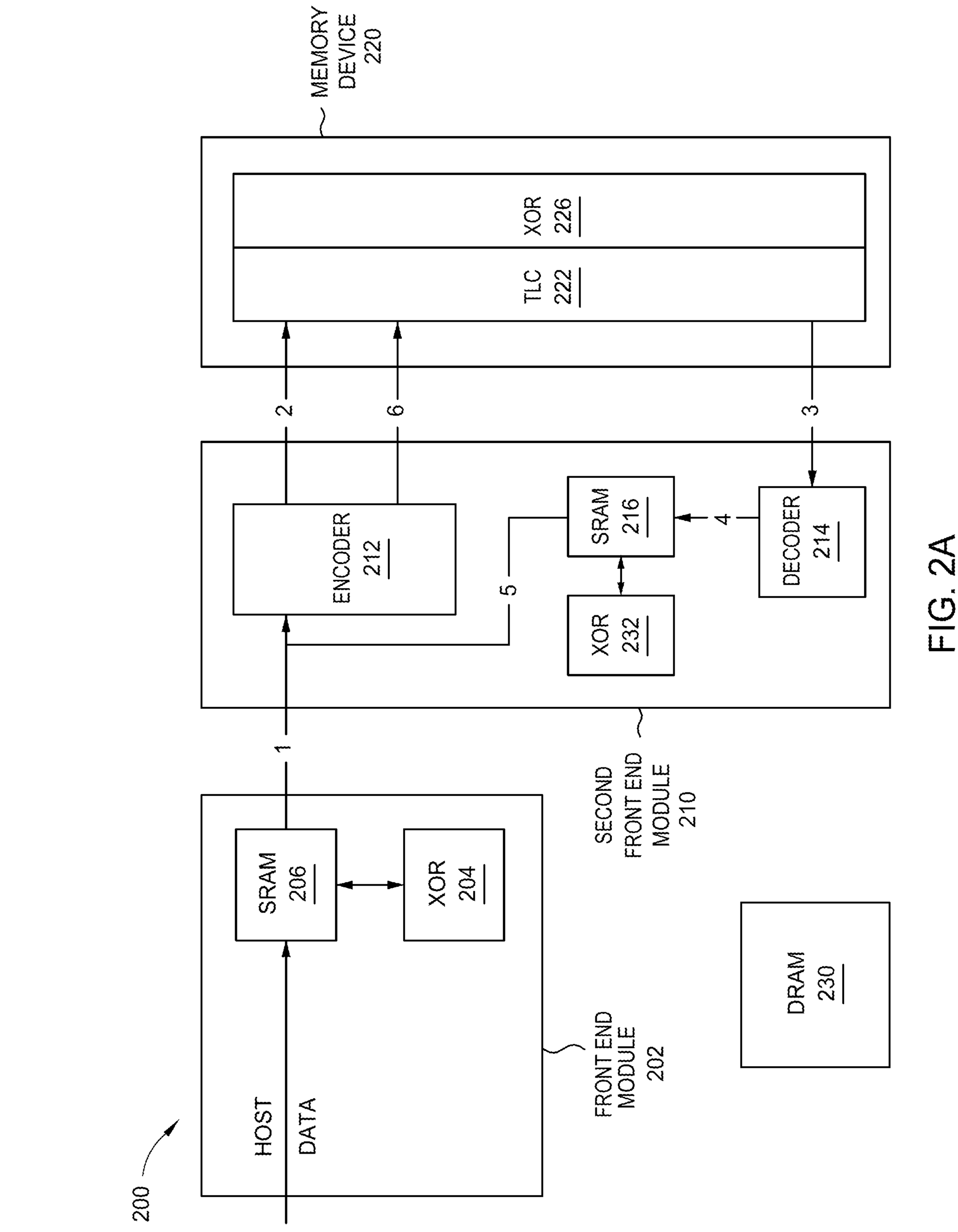

200
HOST
DATA
SRAM 206
XOR 204
FRONT END MODULE 202
SECOND FRONT END MODULE 210
DRAM 230
MEMORY DEVICE 220
TLC 222
XOR 226
ENCODER 212
SRAM 216
XOR 232
DECODER 214
1
2
3
4
5
6

FIG. 2A

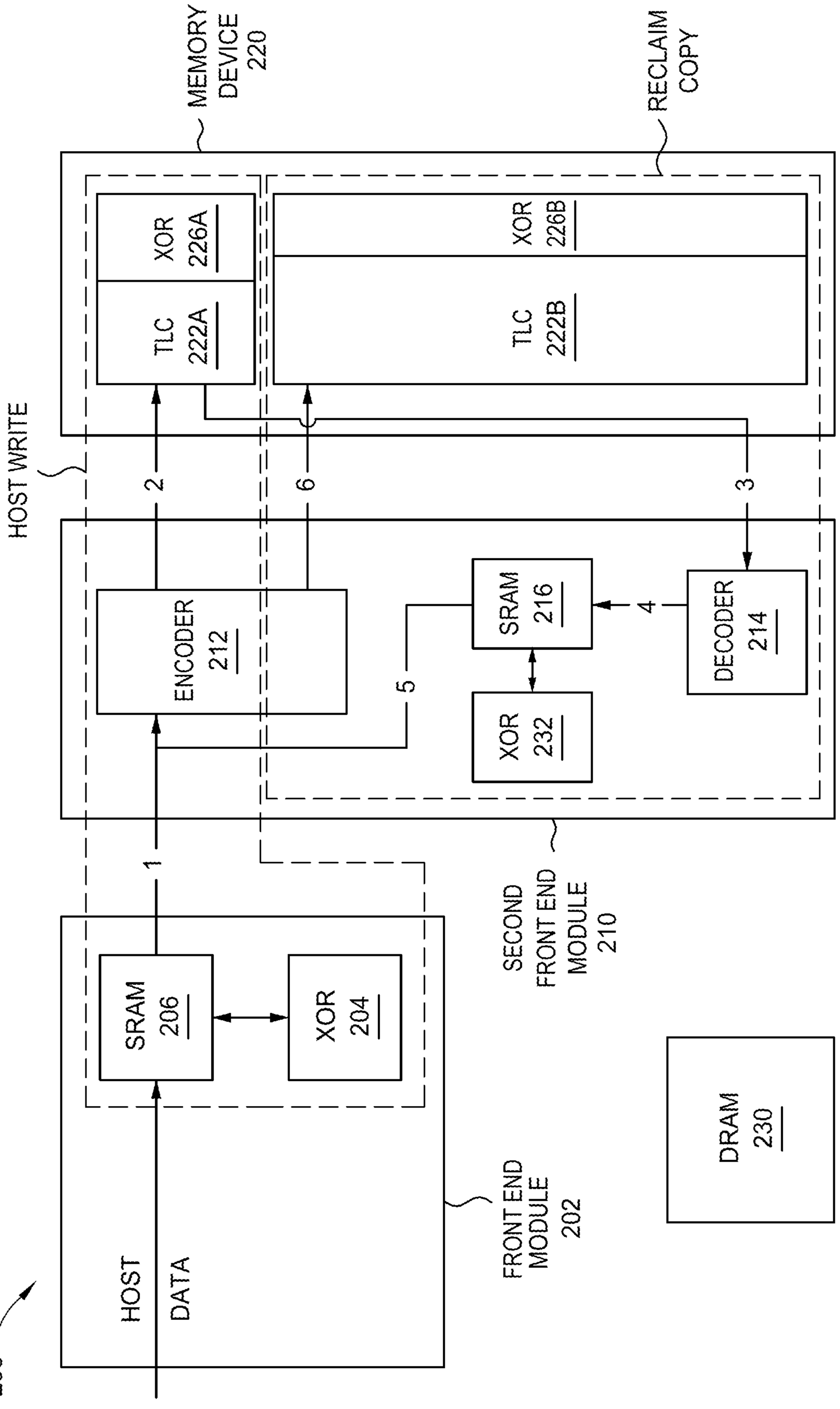

200
HOST
DATA
FRONT END MODULE 202
SRAM 206
XOR 204
SECOND FRONT END MODULE 210
DRAM 230
HOST WRITE
1
2
6
3
MEMORY DEVICE 220
RECLAIM COPY
XOR 226A
TLC 222A
XOR 226B
TLC 222B
ENCODER 212
5
SRAM 216
4
DECODER 214
XOR 232

FIG. 2B

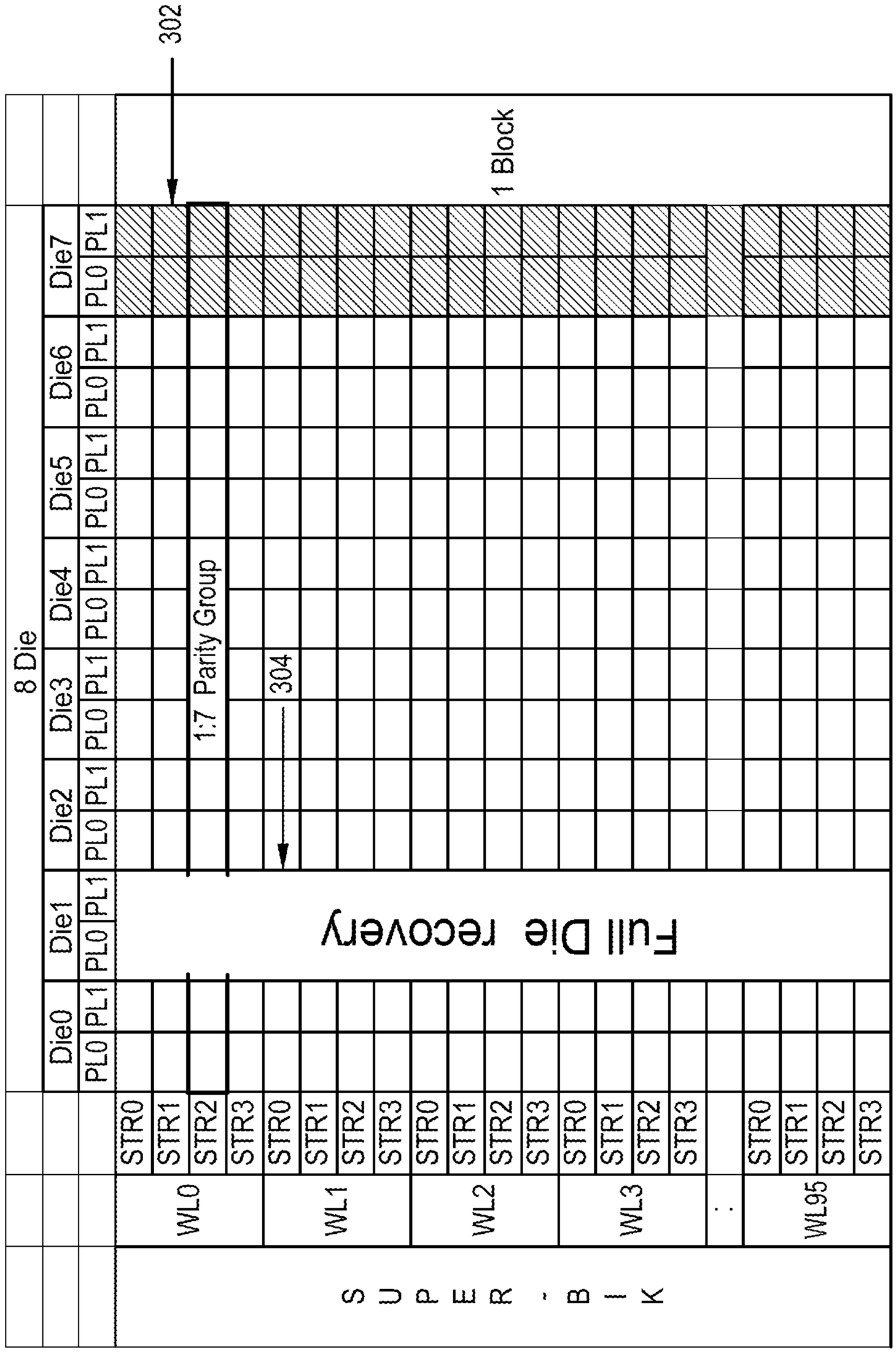

300
SUPER - BLK
WL0
WL1
WL2
WL3
WL95
STR0
STR1
STR2
STR3
Die0
Die1
Die2
Die3
Die4
Die5
Die6
Die7
PL0
PL1
8 Die
1:7 Parity Group
Full Die recovery
1 Block
302
304

FIG. 3

	The errors to be fixed using XOR in NAND							
Source of UECC data errors	PSF program failures	Silent program failures		wear, DR	XOR PARITY			
input metric	BFR/PE cycle			SFR				
category	WL-WL short and smaller	WL-WL short and smaller	1-EB, 2-EB, plane, die	random UECC CWs (SFR)	XOR Parity level	parity ratio examples	Temp data source or XOR holding	Read Verify / EPWR** level
QLC, baseline, Full Die Redundancy (1:31 parity ratio)								
SLC blocks (host writes)	V	V	V	V	FDR	1:31	n/a*	
QLC (copy from SLC or QLC)		V	V	V	FDR	1:31	2WL deep	
QLC ESS drive path (conventional, 32 die super-blocks)								
SLC blocks (host writes)	V	V	V	V	e.g. FDR	1:31		
QLC (copy from SLC or QLC)								
402 → 1. no extra checks		V	V	V	less than SLCs	1:63	2WL deep	
404 → 2. with FW read verify/ EPWR			V	V	even less	1:127	2WL deep	
3. with FW read verify/ EPWR		V		V	even less	1:255	1-EB deep	
4. with FW read verify/ EPWR				V	even less	1:383	1-EB deep	
406 → 5. with FW read verify/ EPWR					none	none	1-EB deep	
EPWR = FW Read Verify - may not detect 100% of errors								
*limited by host buffers, may add extra temp parity (Q-Parity)								
** exact data verification design depends on the failure type and geometry. Smaller error require higher verification overhead.								

FIG. 4

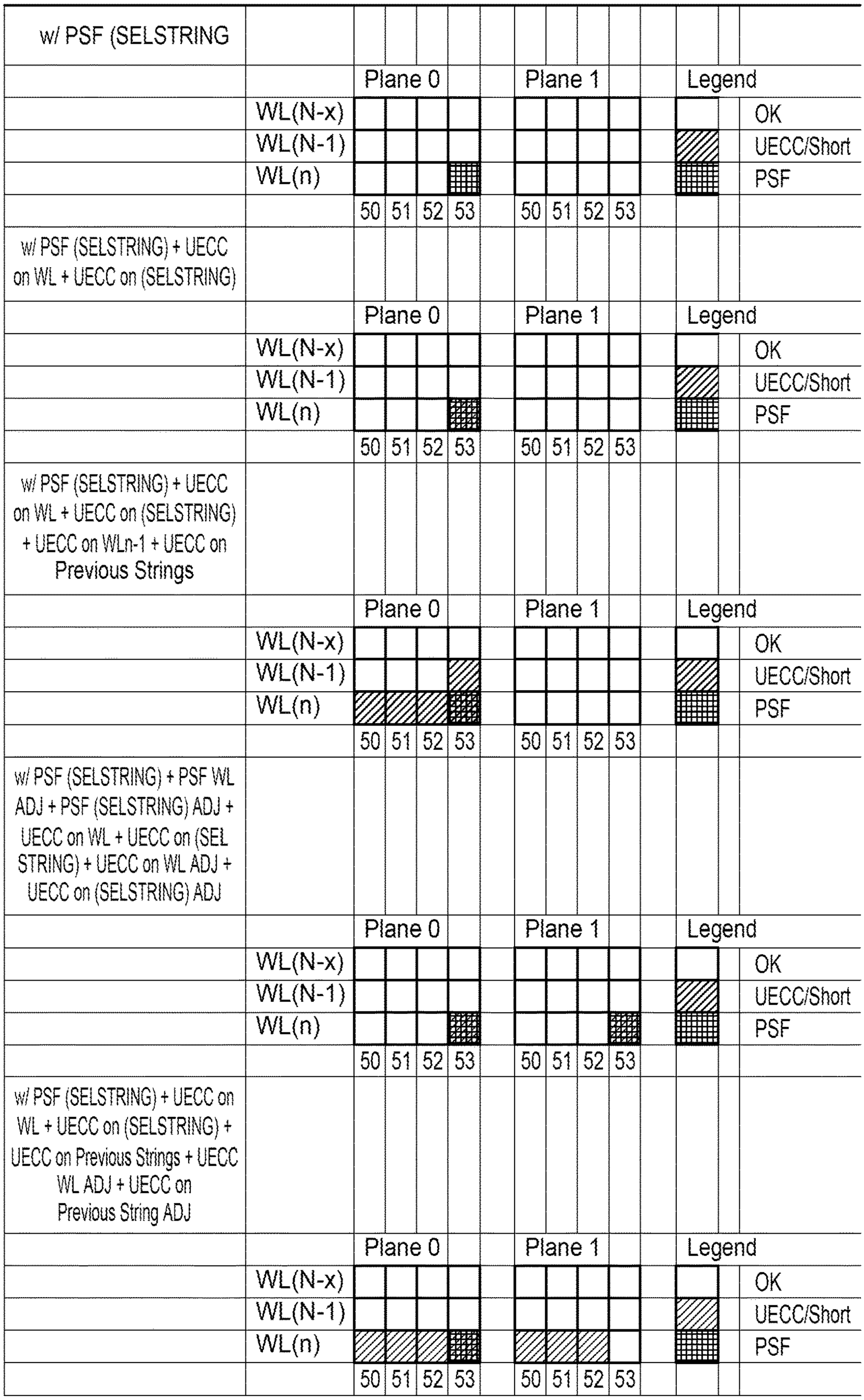

w/ PSF (SELSTRING
Plane 0
Plane 1
Legend
WL(N-x)
WL(N-1)
WL(n)
OK
UECC/Short
PSF
50 51 52 53
w/ PSF (SELSTRING) + UECC on WL + UECC on (SELSTRING)
w/ PSF (SELSTRING) + UECC on WL + UECC on (SELSTRING) + UECC on WLn-1 + UECC on Previous Strings
w/ PSF (SELSTRING) + PSF WL ADJ + PSF (SELSTRING) ADJ + UECC on WL + UECC on (SEL STRING) + UECC on WL ADJ + UECC on (SELSTRING) ADJ
w/ PSF (SELSTRING) + UECC on WL + UECC on (SELSTRING) + UECC on Previous Strings + UECC WL ADJ + UECC on Previous String ADJ

FIG. 5A

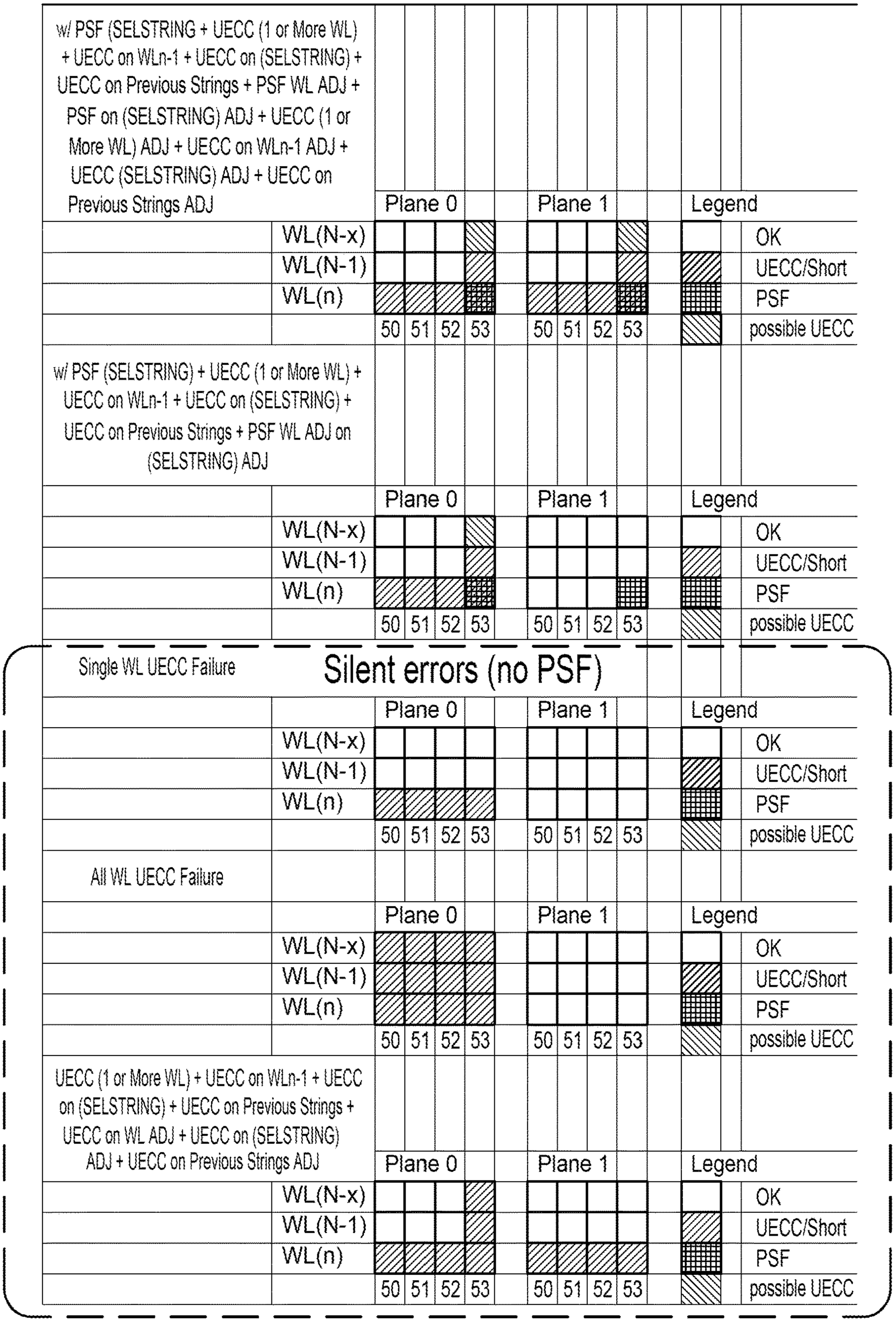

w/ PSF (SELSTRING + UECC (1 or More WL) + UECC on WLn-1 + UECC on (SELSTRING) + UECC on Previous Strings + PSF WL ADJ + PSF on (SELSTRING) ADJ + UECC (1 or More WL) ADJ + UECC on WLn-1 ADJ + UECC (SELSTRING) ADJ + UECC on Previous Strings ADJ
Plane 0
Plane 1
Legend
WL(N-x)
WL(N-1)
WL(n)
50 51 52 53
OK
UECC/Short
PSF
possible UECC
w/ PSF (SELSTRING) + UECC (1 or More WL) + UECC on WLn-1 + UECC on (SELSTRING) + UECC on Previous Strings + PSF WL ADJ on (SELSTRING) ADJ
Single WL UECC Failure
Silent errors (no PSF)
All WL UECC Failure
UECC (1 or More WL) + UECC on WLn-1 + UECC on (SELSTRING) + UECC on Previous Strings + UECC on WL ADJ + UECC on (SELSTRING) ADJ + UECC on Previous Strings ADJ

FIG. 5B

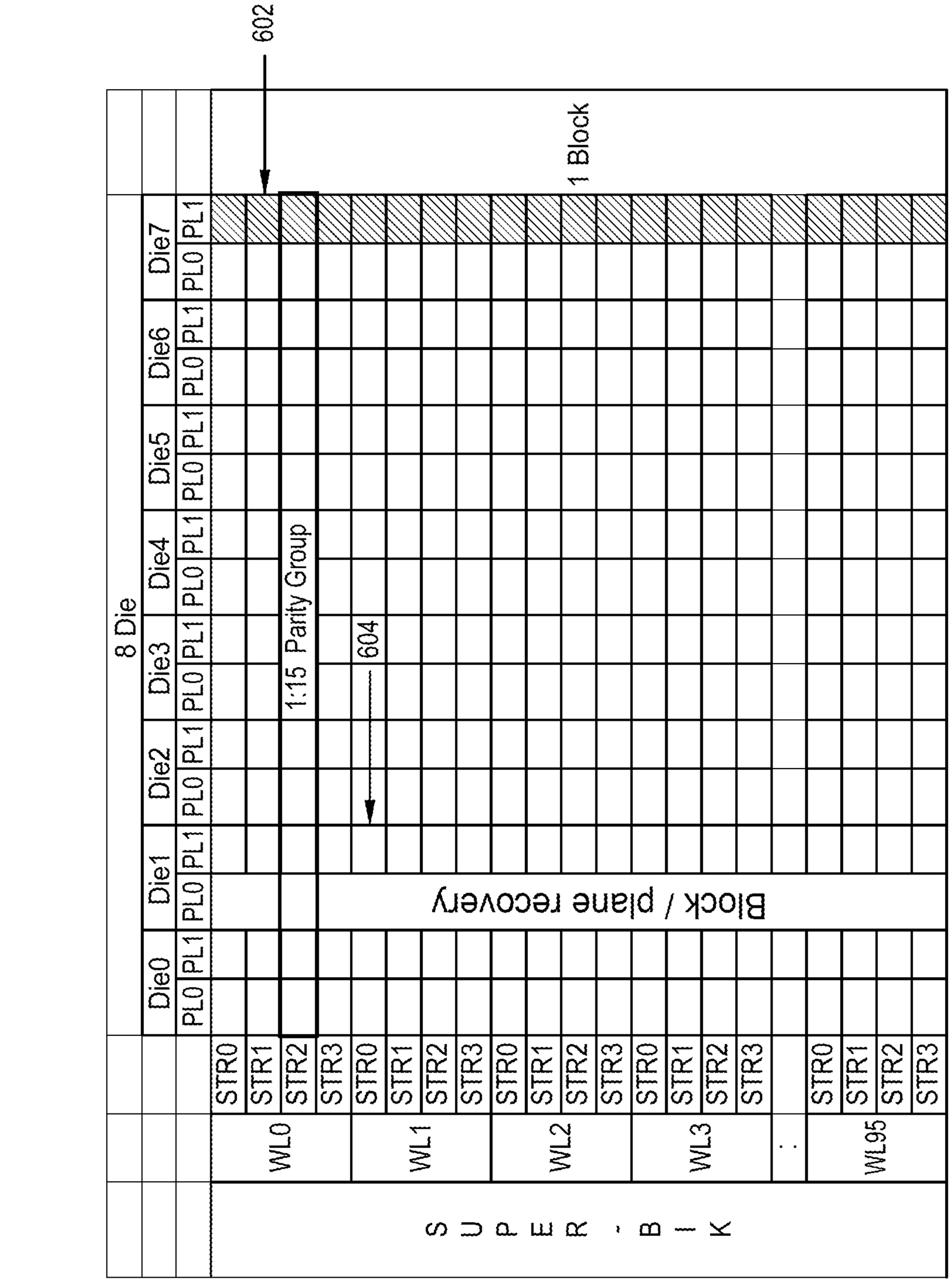

600
SUPER - BLK
WL0
WL1
WL2
WL3
WL95
STR0
STR1
STR2
STR3
Die0
Die1
Die2
Die3
Die4
Die5
Die6
Die7
PL0
PL1
8 Die
1:15 Parity Group
Block / plane recovery
1 Block
602
604

FIG. 6A

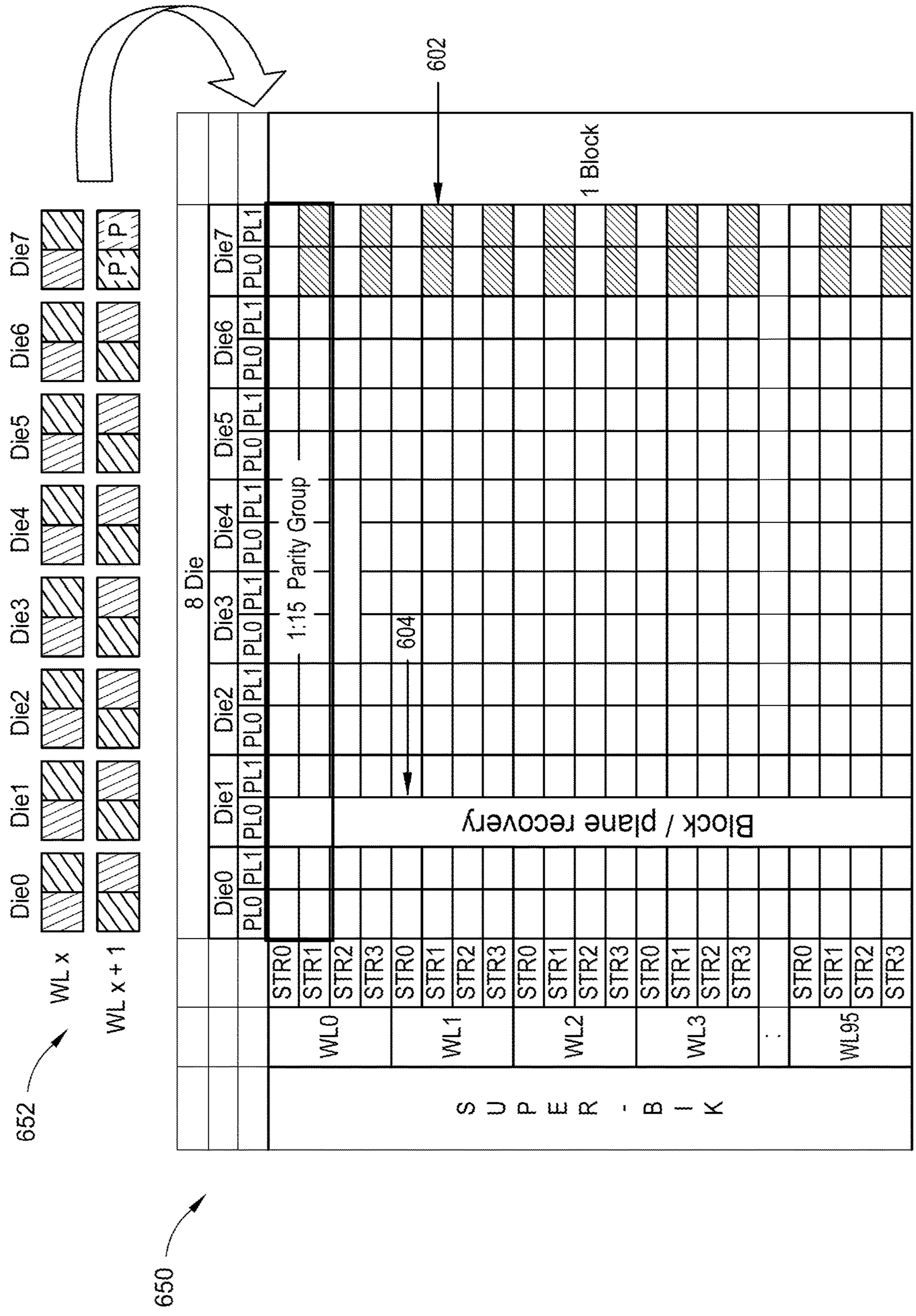

602
1 Block
8 Die
1:15 Parity Group
604
Block / plane recovery
Die7
Die6
Die5
Die4
Die3
Die2
Die1
Die0
PL0
PL1
P
WL x
WL x + 1
652
650
WL0
WL1
WL2
WL3
WL95
STR0
STR1
STR2
STR3
S U P E R - B L K

FIG. 6B

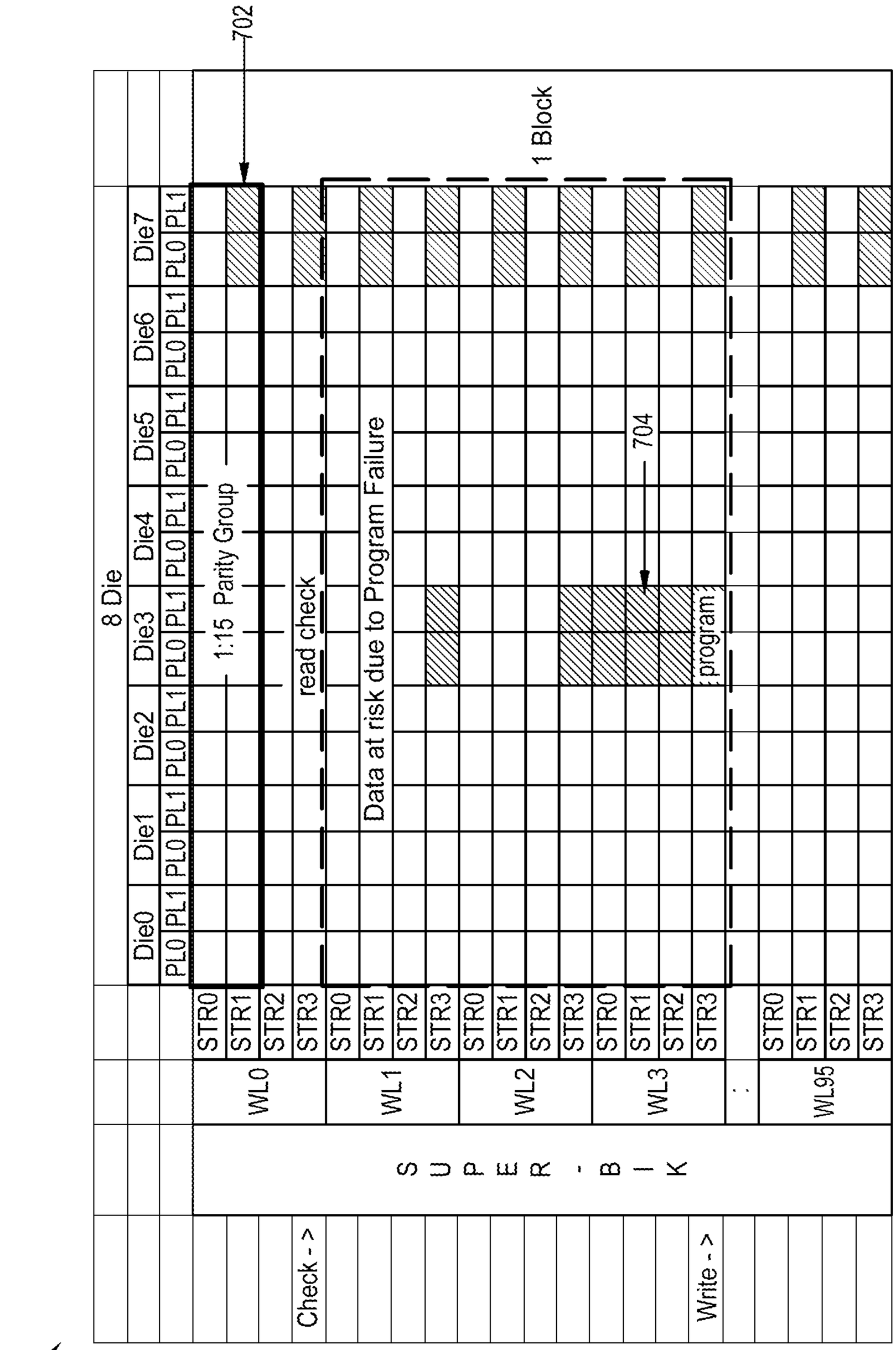

700
702
1 Block
8 Die
Die0
Die1
Die2
Die3
Die4
Die5
Die6
Die7
PL0
PL1
1:15 Parity Group
read check
Data at risk due to Program Failure
704
program
STR0
STR1
STR2
STR3
WL0
WL1
WL2
WL3
WL95
SUPER - BLK
Check - >
Write - >

FIG. 7

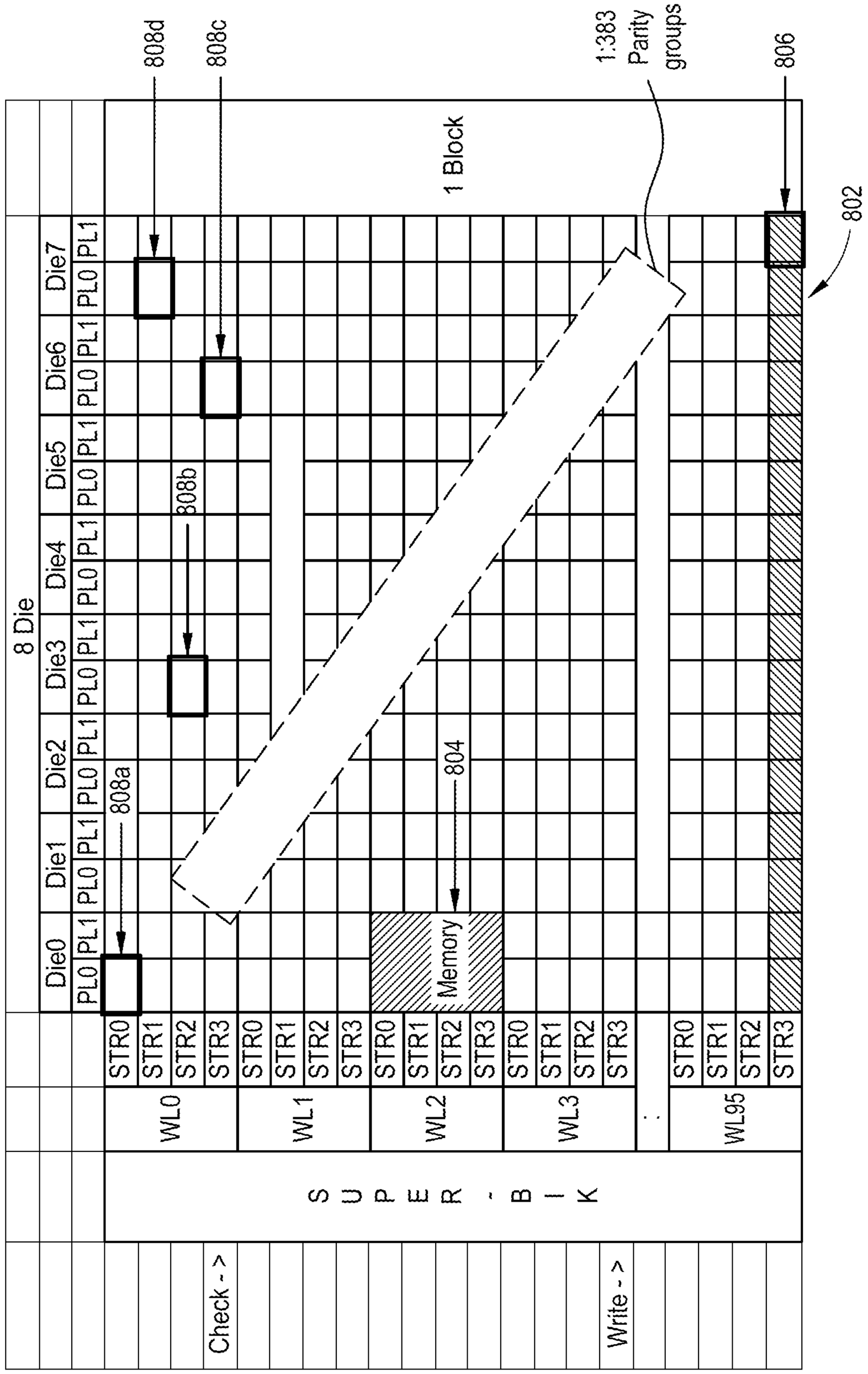

800
8 Die
1 Block
1:383 Parity groups
808d
808c
808b
808a
806
802
804
Memory
Die0
Die1
Die2
Die3
Die4
Die5
Die6
Die7
PL0
PL1
STR0
STR1
STR2
STR3
WL0
WL1
WL2
WL3
WL95
. .
S U P E R - B L K
Check - >
Write - >

FIG. 8

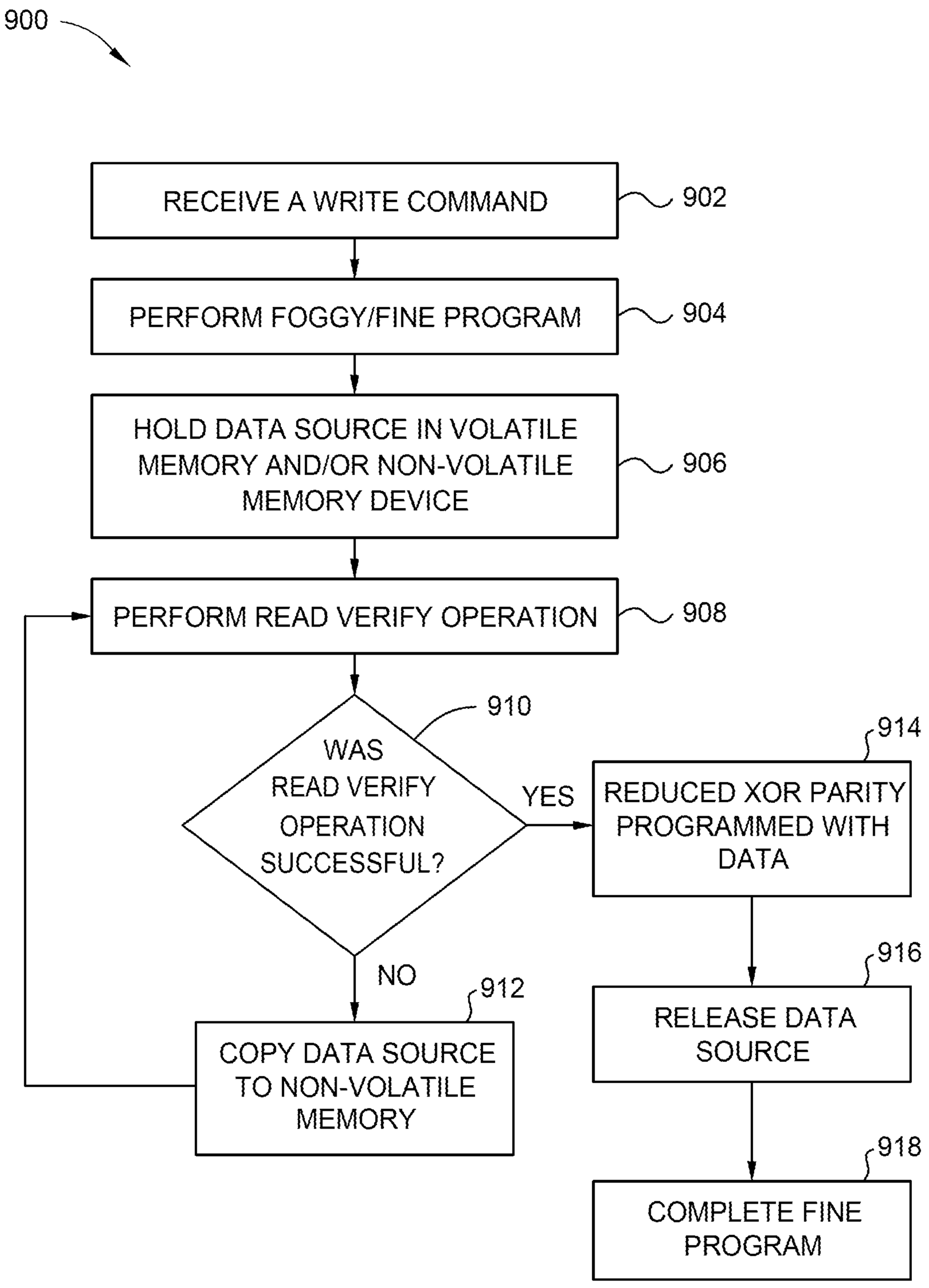

900
RECEIVE A WRITE COMMAND
902
PERFORM FOGGY/FINE PROGRAM
904
HOLD DATA SOURCE IN VOLATILE MEMORY AND/OR NON-VOLATILE MEMORY DEVICE
906
PERFORM READ VERIFY OPERATION
908
WAS READ VERIFY OPERATION SUCCESSFUL?
910
YES
NO
REDUCED XOR PARITY PROGRAMMED WITH DATA
914
COPY DATA SOURCE TO NON-VOLATILE MEMORY
912
RELEASE DATA SOURCE
916
COMPLETE FINE PROGRAM
918

FIG. 9

TLC DATA PROGRAMMING WITH HYBRID PARITY

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a divisional of co-pending U.S. patent application Ser. No. 17/202,163, filed Mar. 15, 2021, now U.S. Pat. No. 11,861,195 which is herein incorporated by reference.

BACKGROUND OF THE DISCLOSURE

Field of the Disclosure

Embodiments of the present disclosure generally relate to improving programming to data storage devices, such as solid state drives (SSDs).

Description of the Related Art

Programming or writing data may require two writing phases: foggy and fine. In foggy-fine programming, the bits to be written cannot simply be written once. Rather, the data needs to be first written by foggy programming where voltage pulses are provided to push the current state to a more resolved state, but not a completely resolved state. Fine programming is performed at a point in time after foggy programming to write the data again in the completely resolved state.

While programing or writing data to the respective locations of the data storage device, program failures, such as bit errors, may accumulate. In order to fix the program failures, exclusive or (XOR) parity data is generated by an error correction engine, such as a parity engine or a low density parity check (LDPC) engine. The XOR parity data is stored in each location of the data write. For example, if data is fine programmed to a first non-volatile memory device, the data is first programmed to a volatile memory device cache or to a second non-volatile memory device cache. The data is then foggy program to the first non-volatile memory device. After completing the foggy program to the first non-volatile memory device, the data is fine programmed to the first non-volatile memory device. XOR parity data may be generated and stored along with the programmed data in each location of the data programming sequence, such as in the volatile memory device cache, the second non-volatile memory device cache, the foggy program to the first non-volatile memory device, and the fine program to the first non-volatile memory device. The accumulation of XOR parity data in each memory device previously mentioned increases the XOR parity overhead and reduces over provisioning or capacity for user data in the data storage device.

Therefore, there is a need in the art for improved programming to the non-volatile memory while decreasing the XOR parity overhead and maintaining or increasing the reliability of the data storage device.

SUMMARY OF THE DISCLOSURE

The present disclosure generally relates to improving programming to data storage devices, such as solid state drives (SSDs). A first memory device has a first XOR element and a second memory device has a second XOR element. The ratio of the first XOR element to the capacity of the first memory device is substantially smaller than the ratio of the second XOR element to the capacity of the second memory device. A read verify operation to find program failures is executed on either a wordline to wordline basis, an erase block to erase block basis, or both a wordline to wordline basis and an erase block to erase block basis. Because the program failures are found and fixed prior to programming to the second memory device, the second XOR element may be decreased substantially.

In one embodiment, a data storage device includes a controller and a memory device coupled to the controller. The memory device includes a first superblock that has a first parity portion of a first storage size and a second superblock that has a second parity portion of a second storage size. The first storage size is less than the second storage size.

In another embodiment, a data storage device includes a memory device and a controller coupled to the memory device. The controller is configured to receive host data from a host device, generate first exclusive or (XOR) parity data for the host data, encode the host data and first XOR parity data with an encoder, write the host data and first XOR parity data to a first memory superblock, decode valid data and first XOR parity data written to the first memory superblock, where the valid data corresponds with the host data that has not been modified by the host device located in the first memory superblock, generate second XOR parity data for the decoded valid data, re-encode the decoded data and second XOR parity data with the encoder, write the re-encoded data and parity data to a second memory superblock. The re-encoded second XOR parity data has a size that is less than the size of the first XOR parity data.

In another embodiment, a data storage device includes memory means comprising a first memory superblock and a second memory superblock, means to store host data with parity data in the first memory superblock, and means to store a copy of the host data and parity data in the second memory superblock, wherein the copy of the parity data utilizes a smaller amount of parity data storage in the second superblock compared to an amount of parity data storage in the first superblock.

BRIEF DESCRIPTION OF THE DRAWINGS

So that the manner in which the above recited features of the present disclosure can be understood in detail, a more particular description of the disclosure, briefly summarized above, may be had by reference to embodiments, some of which are illustrated in the appended drawings. It is to be noted, however, that the appended drawings illustrate only typical embodiments of this disclosure and are therefore not to be considered limiting of its scope, for the disclosure may admit to other equally effective embodiments.

FIG. **1** is a schematic block diagram illustrating a storage system in which data storage device may function as a storage device for a host device, according to disclosed embodiments.

FIGS. **2**A and **2**B are schematic illustrations of scheduling foggy-fine programming, according to disclosed embodiments.

FIG. **3** is a schematic illustration of a horizontal exclusive or (XOR) scheme with full die redundancy of a super block, according to disclosed embodiments.

FIG. **4** is an illustration of the possible options of exclusive or (XOR) in a multi-level cell, according to disclosed embodiments.

FIGS. **5**A and **5**B are illustrations of various program failure types, according to disclosed embodiments.

FIGS. **6**A and **6**B are schematic illustrations of a reduced horizontal exclusive or (XOR) scheme of a superblock, according to disclosed embodiments.

FIG. **7** is a schematic illustration of a reduced horizontal exclusive or (XOR) scheme of a superblock, according to disclosed embodiments.

FIG. **8** is a schematic illustration of a reduced vertical exclusive or (XOR) scheme of a superblock, according to disclosed embodiments.

FIG. **9** is a flowchart illustrating a method of performing a foggy-fine program, according to disclosed embodiments.

To facilitate understanding, identical reference numerals have been used, where possible, to designate identical elements that are common to the figures. It is contemplated that elements disclosed in one embodiment may be beneficially utilized on other embodiments without specific recitation.

DETAILED DESCRIPTION

In the following, reference is made to embodiments of the disclosure. However, it should be understood that the disclosure is not limited to specific described embodiments. Instead, any combination of the following features and elements, whether related to different embodiments or not, is contemplated to implement and practice the disclosure. Furthermore, although embodiments of the disclosure may achieve advantages over other possible solutions and/or over the prior art, whether or not a particular advantage is achieved by a given embodiment is not limiting of the disclosure. Thus, the following aspects, features, embodiments and advantages are merely illustrative and are not considered elements or limitations of the appended claims except where explicitly recited in a claim(s). Likewise, reference to "the disclosure" shall not be construed as a generalization of any inventive subject matter disclosed herein and shall not be considered to be an element or limitation of the appended claims except where explicitly recited in a claim(s).

The present disclosure generally relates to improving programming to data storage devices, such as solid state drives (SSDs). A first memory device has a first XOR element and a second memory device has a second XOR element. The ratio of the first XOR element to the capacity of the first memory device is substantially smaller than the ratio of the second XOR element to the capacity of the second memory device. A read verify operation to find program failures is executed on either a wordline to wordline basis, an erase block to erase block basis, or both a wordline to wordline basis and an erase block to erase block basis. Because the program failures are found and fixed prior to programming to the second memory device, the second XOR element may be decreased substantially.

FIG. **1** is a schematic block diagram illustrating a storage system **100** in which data storage device **106** may function as a storage device for a host device **104**, according to disclosed embodiments. For instance, the host device **104** may utilize a non-volatile memory (NVM) **110** included in data storage device **106** to store and retrieve data. The host device **104** comprises a host DRAM **138**. In some examples, the storage system **100** may include a plurality of storage devices, such as the data storage device **106**, which may operate as a storage array. For instance, the storage system **100** may include a plurality of data storage devices **106** configured as a redundant array of inexpensive/independent disks (RAID) that collectively function as a mass storage device for the host device **104**.

The storage system **100** includes the host device **104**, which may store and/or retrieve data to and/or from one or more storage devices, such as the data storage device **106**. As illustrated in FIG. **1**, the host device **104** may communicate with the data storage device **106** via an interface **114**. The host device **104** may comprise any of a wide range of devices, including computer servers, network attached storage (NAS) units, desktop computers, notebook (i.e., laptop) computers, tablet computers, set-top boxes, telephone handsets such as so-called "smart" phones, so-called "smart" pads, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, or other devices capable of sending or receiving data from a data storage device.

The data storage device **106** includes a controller **108**, NVM **110**, a power supply **111**, volatile memory **112**, an interface **114**, and a write buffer **116**. In some examples, the data storage device **106** may include additional components not shown in FIG. **1** for the sake of clarity. For example, the data storage device **106** may include a printed circuit board (PCB) to which components of the data storage device **106** are mechanically attached and which includes electrically conductive traces that electrically interconnect components of the data storage device **106**, or the like. In some examples, the physical dimensions and connector configurations of the data storage device **106** may conform to one or more standard form factors. Some example standard form factors include, but are not limited to, 3.5" data storage device (e.g., an HDD or SSD), 2.5" data storage device, 1.8" data storage device, peripheral component interconnect (PCI), PCI-extended (PCI-X), PCI Express (PCIe) (e.g., PCIe ×1, ×4, ×8, ×16, PCIe Mini Card, MiniPCI, etc.). In some examples, the data storage device **106** may be directly coupled (e.g., directly soldered) to a motherboard of the host device **104**.

The interface **114** of the data storage device **106** may include one or both of a data bus for exchanging data with the host device **104** and a control bus for exchanging commands with the host device **104**. The interface **114** may operate in accordance with any suitable protocol. For example, the interface **114** may operate in accordance with one or more of the following protocols: advanced technology attachment (ATA) (e.g., serial-ATA (SATA) and parallel-ATA (PATA)), Fibre Channel Protocol (FCP), small computer system interface (SCSI), serially attached SCSI (SAS), PCI, and PCIe, non-volatile memory express (NVMe), OpenCAPI, GenZ, Cache Coherent Interface Accelerator (CCIX), Open Channel SSD (OCSSD), or the like. The electrical connection of the interface **114** (e.g., the data bus, the control bus, or both) is electrically connected to the controller **108**, providing electrical connection between the host device **104** and the controller **108**, allowing data to be exchanged between the host device **104** and the controller **108**. In some examples, the electrical connection of the interface **114** may also permit the data storage device **106** to receive power from the host device **104**. For example, as illustrated in FIG. **1**, the power supply **111** may receive power from the host device **104** via the interface **114**.

The NVM **110** may include a plurality of memory devices or memory units. NVM **110** may be configured to store and/or retrieve data. For instance, a memory unit of NVM **110** may receive data and a message from the controller **108** that instructs the memory unit to store the data. Similarly, the memory unit of NVM **110** may receive a message from the controller **108** that instructs the memory unit to retrieve data. In some examples, each of the memory units may be referred to as a die. In some examples, a single physical chip may include a plurality of dies (i.e., a plurality of memory units).

In some examples, each memory unit may be configured to store relatively large amounts of data (e.g., 128 MB, 256 MB, 512 MB, 1 GB, 2 GB, 4 GB, 8 GB, 16 GB, 32 GB, 64 GB, 128 GB, 256 GB, 512 GB, 1 TB, etc.).

In some examples, each memory unit of NVM **110** may include any type of non-volatile memory devices, such as flash memory devices, phase-change memory (PCM) devices, resistive random-access memory (ReRAM) devices, magnetoresistive random-access memory (MRAM) devices, ferroelectric random-access memory (F-RAM), holographic memory devices, and any other type of non-volatile memory devices.

The NVM **110** may comprise a plurality of flash memory devices or memory units. NVM Flash memory devices may include NAND or NOR based flash memory devices and may store data based on a charge contained in a floating gate of a transistor for each flash memory cell. In NVM flash memory devices, the flash memory device may be divided into a plurality of dies, where each die of the plurality of dies includes a plurality of blocks, which may be further divided into a plurality of pages. Each block of the plurality of blocks within a particular memory device may include a plurality of NVM cells. Rows of NVM cells may be electrically connected using a word line to define a page of a plurality of pages. Respective cells in each of the plurality of pages may be electrically connected to respective bit lines. Furthermore, NVM flash memory devices may be 2D or 3D devices and may be single level cell (SLC), multi-level cell (MLC), triple level cell (TLC), or quad level cell (QLC). The controller **108** may write data to and read data from NVM flash memory devices at the page level and erase data from NVM flash memory devices at the block level.

The data storage device **106** includes a power supply **111**, which may provide power to one or more components of the data storage device **106**. When operating in a standard mode, the power supply **111** may provide power to one or more components using power provided by an external device, such as the host device **104**. For instance, the power supply **111** may provide power to the one or more components using power received from the host device **104** via the interface **114**. In some examples, the power supply **111** may include one or more power storage components configured to provide power to the one or more components when operating in a shutdown mode, such as where power ceases to be received from the external device. In this way, the power supply **111** may function as an onboard backup power source. Some examples of the one or more power storage components include, but are not limited to, capacitors, supercapacitors, batteries, and the like. In some examples, the amount of power that may be stored by the one or more power storage components may be a function of the cost and/or the size (e.g., area/volume) of the one or more power storage components. In other words, as the amount of power stored by the one or more power storage components increases, the cost and/or the size of the one or more power storage components also increases.

The data storage device **106** also includes volatile memory **112**, which may be used by controller **108** to store information. Volatile memory **112** may include one or more volatile memory devices. In some examples, the controller **108** may use volatile memory **112** as a cache. For instance, the controller **108** may store cached information in volatile memory **112** until cached information is written to non-volatile memory **110**. As illustrated in FIG. **1**, volatile memory **112** may consume power received from the power supply **111**. Examples of volatile memory **112** include, but are not limited to, random-access memory (RAM), dynamic random access memory (DRAM), static RAM (SRAM), and synchronous dynamic RAM (SDRAM (e.g., DDR1, DDR2, DDR3, DDR3L, LPDDR3, DDR4, LPDDR4, and the like)).

The data storage device **106** includes a controller **108**, which may manage one or more operations of the data storage device **106**. For instance, the controller **108** may manage the reading of data from and/or the writing of data to the NVM **110**. In some embodiments, when the data storage device **106** receives a write command from the host device **104**, the controller **108** may initiate a data storage command to store data to the NVM **110** and monitor the progress of the data storage command. The controller **108** may determine at least one operational characteristic of the storage system **100** and store the at least one operational characteristic to the NVM **110**. In some embodiments, when the data storage device **106** receives a write command from the host device **104**, the controller **108** temporarily stores the data associated with the write command in the internal memory or write buffer **116** before sending the data to the NVM **110**.

FIG. **2**A is a schematic illustration of scheduling foggy-fine programming, according to disclosed embodiments. The Front End (FE) module **202** comprises a first XOR engine **204** and a first static random-access memory (SRAM) **206**. Host data may be initially delivered to the FE module **202**. The data passes through the first XOR engine **204** and is written to the first SRAM **206**. The first XOR engine **204** generates XOR parity information prior to writing to the first SRAM **206**. Exclusive OR (XOR) parity information is used to improve reliability of storage device for storing data, such as enabling data recovery of failed writes or failed reads of data to and from NVM or enabling data recovery in case of power loss. The storage device may be the data storage device **106** of FIG. **1**. The reliability may be provided by using XOR parity information generated or computed based on data stored to storage device. The first XOR engine **204** may generate a first parity stream to be written to the first SRAM **206**. The first SRAM **206** may contain a plurality of dies to which data may be written.

The Second Flash Manager (FM2) module **210** comprises of an encoder **212**, a second SRAM **216**, a decoder **214**, and a second XOR engine **232**, where the second XOR engine **232** is configured to generate a second parity stream to be written to the second SRAM **216**. The decoder **214** may comprise a low gear (LG) decoder and a high gear (HG) decoder. The LG decoder can implement low power bit flipping algorithms, such as a low density parity check (LDPC) algorithm. The LG decoder may be operable to decode data and correct bit flips where such data has a low bit error rate (BER). The HG decoder can implement full power decoding and error correction algorithms, which may be initiated upon a failure of the LG decoder to decode and correct bit flips in data. The HG decoder can be operable to correct bit flips where such data has a high BER. Alternatively, FM2 may be replaced with a combined FE-FM monochip.

The encoder **212** and decoder **214** (including the LG decoder and HG decoder) can include processing circuitry or a processor (with a computer-readable medium that stores computer-readable program code (e.g., firmware) executable by the processor), logic circuitry, an application specific integrated circuit (ASIC), a programmable logic controller, an embedded microcontroller, a combination thereof, or the like, for example. In some examples, the encoder **212** and the decoder **214** are separate from the storage controller, and in other examples, the encoder **212** and the decoder **214** are embedded in or part of the storage controller. In some

examples, the LG decoder is a hardened circuit, such as logic circuitry, an ASIC, or the like. In some examples, the HG decoder can be a soft decoder (e.g., implemented by a processor). Data may be written to second SRAM **216** after being decoded at the decoder **214**. The data at the second SRAM **216** may be further delivered to the encoder **212**, as discussed below.

The memory device **220** may be a NAND memory device. The memory device **220** may comprise TLC memory **222**. It is to be understood that the embodiments discussed herein may not be limited to TLC memory and may be applicable to any multilevel cell memory such as MLC memory, QLC memory, or the like. SLC memory, MLC memory, TLC memory, QLC memory, and PLC memory are named according to the number of bits that a memory cell may accept. For example, SLC memory may accept one bit per memory cell and QLC memory may accept four bits per memory cell. Each bit is registered on the storage device as a 1 or a 0.

Furthermore, the TLC memory **222** includes a TLC exclusive or (XOR) partition **226**, where the TLC XOR partition **226** stores parity or XOR data. Host data is written to the first SRAM **206** of the FE module **202**. First XOR parity data may be generated, concurrently, at the first XOR engine **204** of the FE module **202** as the host data is written to the first SRAM **206**. The host data and the generated first XOR parity data passes from the first SRAM **206** to the encoder **212** to be encoded along stream **1**. The host data is encoded and foggy written to the TLC memory **222**. Likewise, the generated first XOR parity data is encoded and foggy written to the TLC XOR partition **226** of the memory device **220** along stream **2**. During the foggy write, the controller may selectively choose data to read in order to allow for data sorting into the relevant one or more streams. The TLC memory **222** may be a region of the memory device **220** dedicated to protecting data in case of a power loss event.

At stream **3**, the host data is read from the TLC memory **222** at the decoder **214**. After the host data is decoded at the decoder **214**, the host data is written to the second SRAM **216** of the FM2 **210** along stream **4**, where second XOR parity data is further generated for the host data at the second XOR engine **232** of the FM2 **210**. The host data and second XOR parity data are passed through the encoder **212** to be encoded along stream **5** and are fine written to the respective locations of the TLC memory **222** and the TLC XOR partition **226** along stream **6**.

FIG. **2**B is a schematic illustration of scheduling foggy-fine programming, according to disclosed embodiments. The TLC memory **222** may be further partitioned into a first TLC memory partition **222**A and a second TLC memory partition **222**B. The second TLC memory partition **222**B may be larger than the first TLC memory partition **222**A. Furthermore, the TLC XOR partition **226** may be further partitioned into a first TLC XOR partition **226**A and a second TLC XOR partition **226**B. The second TLC XOR partition **226**B may be smaller than the first TLC XOR partition **226**B. In one embodiment, the size of the second TLC XOR partition **226**B is about 50% of the size of the size of the first TLX XOR partition **226**A. In another example, the size of the second TLC XOR partition **226**B is less than about 50% of the size of the size of the first TLX XOR partition **226**A.

In some examples, the second TLC XOR partition **226**B may be a configurable size, such that the controller **108** may determine the size of the second TLC XOR partition **226**B during the life of the data storage device **106** depending on factors such as a threshold value, TLC memory **222** health, such as the failed bit count (FBC) of the TLC memory, and the like. The first TLC memory partition **222**A and the first TLC XOR partition **226**A may be a first TLC superblock and the second TLC memory partition **222**B and the second TLC XOR partition **226**B may be a second superblock.

When foggy programming to the TLC memory **222**, the host data is programmed to the first TLC XOR partition **222**A. Likewise, the XOR parity data is programmed to the first TLC XOR partition **226**A. The first TLC memory partition **222**A and the first TLC XOR partition **226**A may be considered as a "Host Write" area, where data programmed in the "Host Write" area may be lost or corrupted. However, when fine programming to the TLC memory **222**, the host data is copied (i.e., a reclaimed copy), that is still valid (i.e., has not been re-written or trimmed by another host command), from the first TLC XOR partition **222**A, re-encoded, and programmed to the second TLC memory partition **222**B and the XOR parity data is programmed to the second TLC XOR partition **226**B. After fine programming to the second TLC memory partition **222**B and the second TLC XOR partition **226**B, the data and XOR parity data is protected as a first copy of the data is stored in the first TLC memory partition **222**A. The second TLC memory partition **222**B and the second TLC XOR partition **226**B may be considered as a "Reclaim Copy" area for the reasons stated above.

Because the fine programming may utilize a decreased XOR parity scheme, such as the schemes described below, the required XOR parity partition size may be decreased. For example, a reduction of the XOR parity data of about 50% may release about 100 GB of capacity in a 7680 GB SSD. The 100 GB of capacity released may then be utilized to store additional host data. Thus, over-provisioning of the memory for XOR parity data may be decreased and the memory that would have been used to store XOR parity data may be utilized to store host data or similar data.

FIG. **3** is a schematic illustration of a horizontal exclusive or (XOR) scheme with full die redundancy of a superblock **300**, according to disclosed embodiments. The superblock **300** includes a plurality of dies (e.g., dies 0-7) and a plurality of wordlines (WL) (e.g., WLs 0-95). The listed number of dies and wordlines are not intended to be limiting, but are shown to exemplify a possible embodiment. For example, the superblock may include about 32 dies and more than or less than about 96 WLs. Each die of the plurality of dies includes a first plane, indicated by PL0, and a second plane, indicated by PL1. Furthermore, each of wordline of the plurality of wordlines includes four strings (STR). The number of strings per wordline is based on the type of memory cell of the superblock. For example, QLC memory includes four strings per wordline, TLC memory includes three strings per wordline, and SLC memory includes one string per wordline.

The superblock **300** may be an example of a zone namespace architecture that includes seven dies for data and an eighth die for XOR parity data. Die 7 of the superblock **300** is associated with XOR parity data **302**. Because die 7 includes only XOR parity data **302**, the XOR parity data **302** may recover another failed die **304**, such as die 1, where recovering the another failed die **304** includes recovering all of the data of the failed die (i.e., full die redundancy). Furthermore, because each string, such as string 2 of WL 0, spans across each of the eight dies, each string includes 16 planes. Because die 7 includes XOR parity data **302**, the parity group ratio is about 1:7, where the XOR parity data

overhead is about 12.5% (i.e., ⅛). The listed values are not intended to be limiting, but to provide an example of possible embodiment.

FIG. **4** is an illustration of the possible options of exclusive or (XOR) in a multi-level cell, according to disclosed embodiments. Sources of uncorrectable error correction code (UECC) data errors includes program status failures (PSF), silent program failures, and wear and data retention related random failures. The PSF and the silent program failures may be associated with the bit failure rate (BFR) and/or the program erase (PE) cycle. The wear and DR related random failures may be associated with the sector failure rate (SFR).

The "XOR parity" column illustrates the type of XOR parity included in each embodiment. For example, a full die redundancy (FDR) may recover an entire failed die. However, the XOR parity data associated with the FDR may require a large amount of storage space in the superblock, thus reducing the amount of data that may be stored in the superblock. XOR parity data or any other parity schemes for multiple error correction code (ECC) codewords, such as low-density parity-check (LDPC), may be used to recover failed bits of data. For example, when the failed bit count (FBC) is larger than a threshold value, the controller, such as the controller **108** of FIG. **1**, may utilize the XOR parity data to recover the failed bits, such that the data associated with the failed bits is no includes the failed bits.

Another example of a data error is a program failure, where the program failure size varies from one WL-string-plane to 2-erase block failure two planes. Unlike the FBC errors, where both XOR parity data and LDPC may be used to correct the FBC errors, XOR parity data and similar parity schemes may be used to correct program failures. Program failures, such as PSF, may be fixed by writing data in a different location of the superblock. For example, when a cell has an unacceptable bit error rate (UBER), the controller may avoid programming data to the cell that has an UBER. However, silent program failures may be undetectable by the controller and are passed to the NVM, such as the NVM **110** of FIG. **1**, unnoticed. Silent program failures may result in double, triple, or higher errors that reduce the reliability of the data storage device, such as the data storage device **106** of FIG. **1**.

In order to protect user data from failure due to the program failures, the XOR parity data scheme needs to be large enough (i.e., low XOR parity ratio) to protect against any combinations of the program failures previously described and any program failures not described, but contemplated, and have the correct geometry as well. However, the XOR parity data scheme size has limitations. For example, by increasing the size of the XOR parity data scheme, less user data or any other data may be stored in the NVM since the XOR parity data takes up more memory in the NVM, where the more memory could be utilized to store more user data.

The "extra measures" column refers to the temporary storage of data in a cache or a buffer or the level of read verify/enhanced post write read (EPWR) used to check for and fix errors. For example, the buffer or the cache may store up to the last two wordlines (2WL deep) of data written to the superblock. When another wordline is written to the superblock, the oldest wordline of the last two wordlines stored in the buffer or the cache is released, such that releasing the wordline refers to erasing the data. Furthermore, the read verify/EPWR level refers to the frequency of the read verify/EPWR operations. For example, a read verify/EPWR level of a wordline signifies that after a wordline is programmed, the read verify/EPWR operation occurs.

As illustrated in FIG. **4**, each time there is an additional firmware (FW) read verify/EPWR check, the XOR parity ratio decreases. For example, at the first FW read verify/EPWR check embodiment **404**, where the data is copied from a first TLC memory partition, such as the first TLC memory partition **222**A of FIG. **2**B, to a second TLC memory partition, such as the second TLC memory partition **222**B of FIG. **2**B, the XOR parity ratio is about to about 1:127. As a comparison, the no extra checks embodiment **402** has an XOR parity ratio of about 1:63. The first FW read verify/EPWR check embodiment **404** includes a read verify/EPWR level of a wordline. Furthermore, the cache or the buffer stores the last two wordlines (e.g., 2WL deep) of data written to the superblock. By further performing read verify/EPWR checks, such as the fourth FW read verify/EPWR check embodiment **406**, the XOR parity ratio may be substantially decreased to about 0, where the buffer or the cache stores the last erase block written to the super block and the read verify/EPWR operation occurs after each time a wordline, an erase block, or both a wordline and an erase block are programmed to.

The XOR parity data ratio as well as overhead may be reduced if the PSF does not need to be corrected. For example, if the data, such as up to the last two wordlines written or the last erase block programmed, is still available in the source blocks, such as the SLC memory, the buffer, and/or the cache, the data in the source blocks may be programmed over the PSF failures. In another example, a temporary XOR parity data may be stored in the buffer, the cache, and/or the volatile memory, such as the volatile memory **112** of FIG. **1**, where, in the case of a PSF, one XOR parity element (i.e., XOR parity data) per cached XOR stripe (i.e., one die, last two wordlines) may be fixed.

Furthermore, the XOR parity data ratio as well as overhead may be reduced if the silent program failures do not need to be fixed. For example, if the data, such as up to the last two wordlines written or the last erase block programmed, is still available in the source blocks, such as the first TLC memory partition **222**A, the buffer, and/or the cache, the data in the source blocks may be programmed over the PSF failures. Furthermore, if additional FW read verify/EPWR operations detect the silent program failures by looking for the error signatures for the silent program failure types, the XOR parity ratio and overhead may be reduced. Referring to FIG. **2**B, the XOR parity ratio of the first TLC memory partition **222**A and the first TLC XOR partition **226**A may be about 1:7, where for every 1 die of XOR parity data, 7 dies are appropriated to data. Likewise, the XOR parity ratio of the second TLC memory partition **226**A and the second TLC XOR partition **226**B may be about 1:7. However, by performing additional FW read verify/EPWR operations on the data programed to the second TLC memory partition **226**B, the XOR parity ratio of the second TLC memory partition **222**B and the second TLC XOR partition **226**B may be about 1:15, about 1:31, about 1:63, about 1:127, about 1:255, about 1:383, or the like. It is contemplated that the XOR parity ratio of the second TLC memory partition **222**B and the second TLC XOR partition **226**B may be about 0 where no XOR parity data is stored in the second TLC XOR partition **226**B.

For example, each wordline may be checked for silent program failures when written to a superblock of the second TLC memory partition **222**B. In some embodiments, each plane and/or each string of the wordline is also checked for

silent program failures. Though the overhead of the operation may be large, silent program failures do not get programmed to the second TLC memory partition **222**B unnoticed. Furthermore, because each wordline is checked, only a minimal number of wordlines, such as up to about two wordlines, may be stored in the buffer, the cache, and/or the volatile memory. Thus, the latency of copying the wordlines from the stored location to the second TLC memory partition **222**B or the latency of releasing the wordlines from the stored location may be negligible or small.

In another example, the FW read verify/EPWR operation may check for whole erase block failures. When checking for whole erase block failures, only a few wordlines need to be checked at the end of programming the erase block, thus making the overhead of the operation smaller than the overhead of the operation of checking each wordline. In one embodiment, the number of wordlines checked may be about two wordlines, where the about two wordlines checked are the last wordlines of the programmed erase block. However, because the erase block is checked at the completion of the erase block program to the second TLC memory partition **222**B, the source blocks associated with the data of the erase block may need to be stored in the first TLC memory partition **222**A, the buffer, and/or the cache. Because storing the source blocks associated with the erase block is larger than storing the last two wordlines programmed to the second TLC memory partition **222**B, the latency of releasing the source block from the relevant location may be larger than the latency of releasing up to about two wordlines.

In yet another example, the FW read verify/EPWR operation may check for both erase block and wordline failures. After the erase block has been programmed to the second TLC memory partition **222**B, each wordline of the erase block is checked for wordline failures. In some embodiments, each plane and/or each string of the wordline is checked for program failures. Because each wordline of the erase block is checked at the completion of the erase block program to the second TLC memory partition **222**B, the source blocks associated with the data of the erase block may need to be stored in the first TLC memory partition **222**A, the buffer, and/or the cache. Because storing the source blocks associated with the erase block is larger than storing the last two wordlines programmed to the second TLC memory partition **222**B, the latency of releasing the source block from the relevant location may be larger than the latency of releasing up to about two wordlines. Though the overhead of the operation may be larger than the previous two examples, the silent program failures may not be passed to the second TLC memory partition **222**B unnoticed, thus enabling the highest reduction of XOR parity data, where the XOR parity data stored may be minimal or substantially about zero. In some embodiments, the only XOR parity data stored may be for wear and DR related random failures.

FIGS. **5**A and **5**B are illustrations of various program failure types, according to disclosed embodiments. It is to be understood that FIGS. **5**A and **5**B are, collectively, representative of a single figure split into two pages labeled FIG. **5**A and FIG. **5**B. The silent errors (i.e., silent program failures) do not include any PSF failures. Thus, the silent errors may be passed to the MLC memory, such as the MLC memory **224** of FIG. **2**. However, as illustrated in the other examples of FIGS. **5**A and **5**B, by including at least one cell that has a PSF, the error may be noticed by the controller and the controller may correct the error utilizing XOR parity data, LDPC, and/or the like. The various program failure types may be UECC/short and PSF. In some embodiments, the program failures are along the borders of the superblock, such as the last wordline programmed (i.e., WL(n)) and/or the last string programmed (i.e., S3). By implementing a read verify/EPWR operation, such as the read verify/EPWR operations described in FIG. **4**, the program failures illustrated in FIGS. **5**A and **5**B may not be programmed to the MLC memory. It is contemplated that other failure types not shown are applicable the embodiments described herein.

FIGS. **6**A and **6**B are schematic illustrations of a reduced horizontal exclusive or (XOR) scheme of a superblock **600**, **650**, according to disclosed embodiments. Aspects of the superblock **300** of FIG. **3** may be similar to the superblock **600** of FIG. **6**A and the superblock **650** of FIG. **6**B. Although a superblock scheme is exemplified, it is contemplated that the disclosed embodiments may be applicable to non-superblock schemes. The superblock scheme may refer to data striping across multiple dies, as to achieve a higher level of parallelism. The non-superblock scheme may refer to data striping across a single die. Furthermore, it is to be understood that while a horizontal XOR scheme is exemplified, the embodiments may be applicable to a reduced vertical XOR scheme. According to the horizontal XOR scheme, the XOR parity group/stripe spans horizontally across a data stripe, such that the last block or set of blocks of the horizontal data stripe is programmed with XOR parity data. The XOR parity data may protect against data failure for the respective horizontal data stripe. According to the vertical XOR parity scheme, the XOR parity group spans vertically across a single plane, for example, such that the last block of a plane or set of blocks of the plane is programmed with XOR parity data. The XOR parity data may protect against data failure for the respective vertical data stripe.

Unlike the superblock **300** of FIG. **3**, the superblock **600**, specifically die 7, has about 50% reduced XOR parity data when compared to the superblock **300**. The XOR parity data **602** may be located on a single plane of a single die, such as PL1 of die 7. Like the superblock **600**, the superblock **650** has about 50% reduced XOR parity data when compared to the superblock **300**. However, rather than having XOR parity data on a single plane of a single die, the XOR parity data **602** may be stored on alternating strings, such as on STR1 and STR3 of WL0, where the XOR parity data **602** is stored on both PL0 and PL1 of a single die, such as the die 7. Likewise, the 1:15 parity group **652** illustrates where the parity (P) may be located on alternating wordlines.

The reduced XOR parity data may be due the additional read verify/EPWR operations to check for PSF and silent program failures. The about 50% XOR parity data of the superblock **600**, **650** may only recover a block or a plane of a die **604** rather than a whole die failure. The parity group ratio may be about 1:15 rather than 1:7 as previously illustrated in FIG. **3**. However, because the additional read verify/EPWR operations are executed on the programmed wordlines and/or erase blocks of the superblock, the UBER may be substantially less than the programmed wordlines and/or erase blocks of the superblock that does not have additional read verify/EPWR operations executed on the wordlines and/or erase blocks of the superblock.

FIG. **7** is a schematic illustration of a reduced horizontal exclusive or (XOR) scheme of a superblock **700**, according to disclosed embodiments. Aspects of the superblock **650** of FIG. **6**B may be similar to the superblock **700**. Although a superblock scheme is exemplified, it is contemplated that the disclosed embodiments may be applicable to non-superblock schemes. Furthermore, it is to be understood that while a vertical XOR scheme is exemplified, the embodi-

ments may be applicable to a reduced horizontal XOR scheme. For example, the superblock **700** has a 1:15 parity group ratio, where the XOR parity data **702** is stored on alternating strings. Regarding the superblock **700**, the fourth string, STR3, of the fourth wordline, WL3 is being programmed to. The data at risk due to program failure are the two previously programmed wordlines WL2 and WL1. However, the source blocks for the data of WL2 and WL1 are stored in the TLC memory, such as the first TLC memory partition **222**A of FIG. **2**B. WL0 may be considered "safe", where a successful read verify/EPWR operation, such as the read verify/EPWR operation described in FIG. **4**, has been completed on the data of WL0. The program failures **704**, both PSF and silent program failures, may still exist in WL1, WL2, and WL3 because the read verify/EPWR operation has not yet been executed.

FIG. **8** is a schematic illustration of a reduced vertical exclusive or (XOR) scheme of a superblock **800**, according to disclosed embodiments. Although a superblock scheme is exemplified, it is contemplated that the disclosed embodiments may be applicable to non-superblock schemes. Furthermore, it is to be understood that while a vertical XOR scheme is exemplified, the embodiments may be applicable to a reduced horizontal XOR scheme. The superblock **800** exemplifies the 1:383 parity groups scheme as illustrated in FIG. **4**, where the 1:383 refers to 1 XOR parity data for 383 other cells in a TLC memory that includes 96 wordlines and 8 dies. In some examples, the number of parity groups per superblock may be reduced to half or a quarter of the superblock, such that the superblock includes between about 2 and about 4 parity groups. For example, each of the parity groups may protect against data failures of the respective die and a neighboring adjacent die.

Rather than programming XOR parity data to the last die of each wordline, the XOR parity data **802** is programmed to the last string of the last wordline, such that the XOR parity data protects the previous wordlines and the previous strings for each plane and/or die. In some examples, the XOR parity data **802** may be stored in volatile memory, such as the volatile memory **112** of FIG. **1**, until the completion of the program to the previous strings and wordlines as to maintain a sequential program to the superblock.

In one example, the XOR parity data may protect along the same die and/or plane, such that a first XOR parity data **806** in Die7, PL1, WL 95, STR3 may protect a fourth location **808***d* in Die7, PL0, WL0, STR1. In another example, the first XOR parity data **806** in Die7, PL1, WL 95, STR3 may protect a scattered group of cells such as the first location **808***a*, second location **808***b*, third location **808***c*, and fourth location **808***d*. Furthermore, volatile memory and/or the NVM, such as the first TLC memory partition **222**A of FIG. **2**B, may store the last erase block of data such that a first erase block **804** may be recovered.

FIG. **9** is a flowchart illustrating a method **900** of performing a foggy-fine program, according to disclosed embodiments. At block **902**, the controller, such as the controller **108** of FIG. **1**, receives a write command. The controller performs a foggy/fine program at block **904** to the non-volatile memory, such as the second TLC memory partition **222**B of FIG. **2**B. The data associated with the fine program may be written to one or more wordlines, such as a first wordline and a second wordline, of the NVM or to an erase block of the NVM.

At block **906**, the data source (i.e., the data associated with the write command at block **902**) is held in a volatile memory, such as the volatile memory **112** of FIG. **1**, and/or in a NVM, such as the first TLC memory partition **222**B of FIG. **2**B, where the data source may store up to about two wordlines last fine program written to the NVM or the last erase block fine programmed to the NVM. At block **908**, a read verify operation occurs on the data that is fine programmed to the NVM. The read verify operation may be an enhanced post write read operation. Furthermore, the read verify operation may occur to either the last two wordlines previously programmed, such as a first wordline and a second wordline, the last erase block previously written, or each wordline of the last erase block previously written. The read verify operation checks for program failures such as PSF, silent program failures, and the like.

At block **910**, the controller determines if the read verify operation was successful. If the read verify operation was not successful (i.e., a program failure is present) at block **910**, then at block **912**, the data source stored in the first TLC memory partition **222**A is copied to the second TLC memory partition **222**B, where copying the data to the second TLC memory partition **222**B is a fine program. The controller then performs a read verify operation on the copied data at block **908**. However, if the read verify operation is successful at block **910**, then a reduced amount of XOR parity data programed with data at block **914**, such that the XOR parity data programmed may be about 50% of the XOR parity data programmed in previous approaches. The amount of XOR parity data programmed may depend on the level of programming redundancy performed, such as the different redundancy levels described previously in FIG. **4**.

For example, when programming the data to the second TLC memory partition **222**B, a plane of a die across each of the plurality of wordlines may include an XOR parity element (i.e., XOR parity data). The plurality of wordlines include a plurality of strings, where the XOR parity element is written to alternating strings of a die and at least one string of the plurality of strings does not include XOR parity data. Rather than programming a whole die with XOR parity data, about half a die is programmed with XOR parity data after a successful read verify operation. It is to be understood that the reduced XOR parity data scheme may be a reduced horizontal parity scheme, a vertical reduced parity scheme, or a combination of the previously mentioned schemes. Furthermore, in some examples the vertical reduced parity scheme may be a scattered vertical parity scheme, such that the XOR parity data does protects different cells or bits per plane of each die. In some embodiments, the first TLC memory partition **222**A and the second TLC memory partition **222**B may not have the same XOR parity scheme. For example, the first TLC memory partition **222**A has a first ratio, where the first ratio is about a 1:7 parity group ratio, and the second TLC memory partition **222**B has a second ratio, where the second ratio is about a 1:15 parity group ratio. Furthermore, in some examples, the second TLC memory partition **222**B may have not have any XOR parity data, such as the fourth FW read verify/EPWR check embodiment **406** described in FIG. **4**.

At block **916**, the data source associated with the data of the successful read verify operation is released from the first TLC memory partition **222**A, the buffer, and/or the cache. At block **918**, the fine program is completed.

By performing a read verify operation to different levels of the non-volatile memory, such as on a wordline to wordline basis, on an erase block basis, or on each wordline of an erase block basis, and storing the data source in the SLC memory, the buffer, and/or the cache, the size of the XOR parity data of the fine program to the NVM may be decreased and the reliability of the data may be increased.

In one embodiment, a data storage device includes a controller and a memory device coupled to the controller. The memory device includes a first superblock that has a first parity portion of a first storage size and a second superblock that has a second parity portion of a second storage size. The first storage size is less than the second storage size.

The first superblock has a third storage size and the second superblock has a fourth storage size. The third storage size is less than the fourth storage size. The first superblock is configured to handle host write data. The second superblock is configured to handle reclaimed copies of the host write data. The first superblock is TLC memory. The second superblock is TLC memory. The first parity portion and the second parity portion are XOR parity. The second storage size is about 50% less than the first storage size. The second superblock includes a plurality of wordlines. Each wordline includes a plurality of strings. At least one string of the plurality of strings does not include exclusive or (XOR) data.

In another embodiment, a data storage device includes a memory device and a controller coupled to the memory device. The controller is configured to receive host data from a host device, generate first exclusive or (XOR) parity data for the host data, encode the host data and first XOR parity data with an encoder, write the encoded host data and first XOR parity data to a first memory superblock, decode valid data and first XOR parity data written to the first memory superblock, wherein the valid data corresponds with the host data that has not become obsolete when located in the first memory superblock, generate second XOR parity data for the decoded valid data, re-encode the decoded valid data and second XOR parity data with the encoder, and write the re-encoded data and second XOR parity data to a second memory superblock. The re-encoded second XOR parity data has a size that is less than the size of the first XOR parity data.

The memory device includes a plurality of dies. Each of the plurality of dies includes a first plane and a second plane. At least one plane of the first plane and the second plane includes exclusive or (XOR) data. The controller is further configured to write data to a first wordline of plurality of wordlines of the memory device, write data to a second wordline of the plurality of wordlines, perform a read verify operation on the first wordline, and perform a read verify operation on the second wordline. At least one of the first wordline and the second wordline does not include an XOR parity element. The read verify operation is enhanced post write read (EPWR). The XOR parity element is at least one of a full die redundancy, a full plane redundancy, and an erase block redundancy. The decoder and the encoder are disposed in a front module of the data storage device. The first XOR parity data is generated in a front end module separate from the front module.

In another embodiment, a data storage device includes memory means comprising a first memory superblock and a second memory superblock, means to store host data with parity data in the first memory superblock, and means to store a copy of the host data and parity data in the second memory superblock, wherein the copy of the parity data utilizes a smaller amount of parity data storage in the second superblock compared to an amount of parity data storage in the first superblock.

The data storage device further includes means to perform a read verify operation to detect program failures. The read verify operation is an enhanced post write read. The means to perform a read verify operation includes either checking each wordline of a plurality of wordlines of the memory means for program failures, each erase block of a plurality of erase blocks of the memory means for program failures, or both each wordline of the plurality of wordlines and each erase block of the plurality of erase blocks for program failures.

While the foregoing is directed to embodiments of the present disclosure, other and further embodiments of the disclosure may be devised without departing from the basic scope thereof, and the scope thereof is determined by the claims that follow.

What is claimed is:

1. A data storage device, comprising:

a controller; and

a memory device coupled to the controller, wherein the memory device comprises:

a first superblock that has a first parity portion of a first storage size; and

a second superblock that has a second parity portion of a second storage size, wherein the first storage size is less than the second storage size.

2. The data storage device of claim **1**, wherein the first superblock has a third storage size and the second superblock has a fourth storage size, and wherein the third storage size is less than the fourth storage size.

3. The data storage device of claim **2**, wherein the first superblock is configured to handle host write data.

4. The data storage device of claim **3**, wherein the second superblock is configured to handle reclaimed copies of the host write data.

5. The data storage device of claim **1**, wherein the first superblock is TLC memory.

6. The data storage device of claim **5**, wherein the second superblock is TLC memory.

7. The data storage device of claim **1**, wherein the first parity portion and the second parity portion are XOR parity.

8. The data storage device of claim **1**, wherein the second storage size is about 50% less than the first storage size.

9. The data storage device of claim **1**, wherein the second superblock comprises a plurality of wordlines, wherein each wordline comprises a plurality of strings, and wherein at least one string of the plurality of strings does not include exclusive or (XOR) data.

10. A data storage device, comprising:

memory means comprising a first memory superblock and a second memory superblock;

means to store host data with parity data in the first memory superblock; and

means to store a copy of the host data and parity data in the second memory superblock, wherein the copy of the parity data utilizes a smaller amount of parity data storage in the second memory superblock compared to an amount of parity data storage in the first memory superblock.

11. The data storage device of claim **10**, further comprising:

means to perform a read verify operation to detect program failures, wherein the read verify operation is an enhanced post write read.

12. The data storage device of claim **11**, wherein the means to perform a read verify operation comprises either checking each wordline of a plurality of wordlines of the memory means for program failures, each erase block of a plurality of erase blocks of the memory means for program failures, or both each wordline of the plurality of wordlines and each erase block of the plurality of erase blocks for program failures.

13. The data storage device of claim 10, wherein the memory means comprises a plurality of dies, and wherein each of the plurality of dies comprises a first plane and a second plane.

14. The data storage device of claim 13, wherein at least one plane of the first plane and the second plane includes exclusive or (XOR) data.

15. The data storage device of claim 13, further comprising a controller configured to:

write data to a first wordline of the memory means;

write data to a second wordline of the memory means;

perform a read verify operation on the first wordline; and

perform a read verify operation on the second wordline, wherein at least one of the first wordline and the second wordline does not include an XOR parity element.

16. The data storage device of claim 15, wherein the read verify operation is enhanced post write read (EPWR).

17. The data storage device of claim 15, wherein the XOR parity element is at least one of a full die redundancy, a full plane redundancy, and an erase block redundancy.

18. The data storage device of claim 10, further comprising a decoder and an encoder disposed in a front module of the data storage device.

19. The data storage device of claim 10, wherein first XOR parity data is generated in a front end module and wherein the device further includes a front module that is separate from the front end module.

20. The data storage device of claim 10, wherein the memory means comprises:

a first superblock that has a first parity portion of a first storage size; and

a second superblock that has a second parity portion of a second storage size, wherein the first storage size is less than the second storage size.

* * * * *